Dynamic Weaving in Aspect Oriented Business Process Management

Amin Jalali[1], Petia Wohed[1], Chun Ouyang[2⋆], and Paul Johannesson[1]

[1] Department of Computer and Systems Sciences, Stockholm University, Sweden
{aj,petia,pajo}@dsv.su.se
[2] Science and Engineering Faculty, Queensland University of Technology, Australia
c.ouyang@qut.edu.au

Abstract. Reducing complexity in Information Systems is an important topic in both research and industry. One strategy to deal with complexity is separation of concerns, which results in less complex, easily maintainable and more reusable systems. Separation of concerns can be addressed through the Aspect Oriented paradigm. Although this paradigm has been well researched in programming, it is still at the preliminary stage in the area of Business Process Management. While some efforts have been made to extend business process modelling with aspect oriented capability, it has not yet been investigated how aspect oriented business process models should be executed at runtime. In this paper, we propose a generic solution to support execution of aspect oriented business process models based on the principle behind dynamic weaving of aspects. This solution is formally specified using Coloured Petri Nets. The resulting formal specification serves as the blueprint to the implementation of a service module in the framework of a state-of-the-art Business Process Management System. Using this developed artefact, a case study is performed in which two simplified processes from real business in the domain of banking are modelled and executed in an aspect oriented manner. Through this case study, we also demonstrate that adoption of aspect oriented modularization increases the reusability while reducing the complexity of business process models in practice.

Keywords: Business Process Management, Aspect Oriented, Weaving, Service Oriented Architecture, Reusability, Coloured Petri Nets

1 Introduction

Reducing the complexity of models is an important issue in the Business Process Management (BPM) area. Business process models tend to become complex quickly [4], which makes them difficult to communicate, use, maintain and validate [28]. Various approaches have been proposed to reduce the complexity of process models (e.g. [15, 18, 32, 33]). Some of these approaches have been analysed and systemised as a collection of patterns [28]. One of the patterns is called *orthogonal modularization*, which aims to reduce the complexity of a model by

⋆ Involvement in this work is supported by an ARC Discovery grant with number DP120101624.

separating different aspects of a process, such as security and privacy. Traditionally, these aspects are defined in a single process model, hence adding to the complexity of the model [34]. In contrast, orthogonal modularization advocates modelling the aspects as separate processes. These processes are related to the main process, where they represent different pieces of the puzzle. The business process is described as a result of putting together all pieces of the puzzle. The mechanism that puts all aspects and the main process model together is called *weaving*, while the whole technique is called *aspect oriented modularization*.

Aspect oriented modularization so far has been realised as extensions to current business process modelling techniques such as Aspect Oriented Business Process Modeling Notation (AO4BPMN) [7, 13, 14, 18]. Existing work on AO4BPEL [12] proposed an aspect-oriented extension to Business Process Execution Language for Web Services (BPEL), and their approach for weaving of apsects was defined for that specific language only. How to support the enactment of aspect oriented business process models in general is still an open issue.

In this paper, we present a solution to runtime execution of aspect oriented process models based on the principle behind dynamic weaving of aspects. It is defined in a generic manner, i.e. independent of any specific business process modelling technique or Business Process Mangement System (BPMS). The proposed solution, in the form of a so-called *Aspect Service*, is formally specified using Coloured Petri Nets (CPNs). We select CPN as it is a widely-used formal technique for system design and verification, and its application in the domain of workflow management has been well-established [1]. The CPN specification of Aspect Service serves as a blueprint for design and implementation of a service module which extends the capability of a state-of-the-art BPMS with support to aspect oriented business process enactment. The developed artefact has been used in a real banking case study to validate our solution. The case study also demonstrates that by adopting aspect oriented modularization one can reduce the complexity while increase the reusability of process models in practice.

The remainder of this paper is structured as follows. Section 2 provides a background of the aspect oriented business process modelling. Section 3 describes an overview of our solution to support weaving of aspects during process enactment. Section 4 presents the formalization of the solution in CPN. This is followed by the description of a supporting implementation within an open source BPMS environment in Section 5. Section 6 describes a case study that is conducted using the implemented artefact. Section 7 discusses related work, and finally Section 8 concludes the paper and outlines directions for future work.

2 Background

Process models encompass different activities, which address different concerns of business processes. Charfi et al. enumerate compliance, auditing, business monitoring, accounting, billing, authorization, privacy and separation of duties as examples of concerns [13]. It is common that some of these concerns are scattered across several business processes.

As a real example in Swedish public organizations, it is compulsory to inform citizens if a decision is made on their applications. Accordingly, an `inform` activ-

ity is required in all business processes that contain a `decide` activity. Moreover, a process may contain several `decide` activities, implying the need for several `inform` activities. If the `inform` activity is changed, or if the policy regarding the informing concern is modified, we have to find and update all business processes containing a `decide` activity. To be conformed to the law, when designing a new business process one has to remember to add the `inform` activity after each `decide` activity. Such efforts add costs in designing, updating and monitoring business processes, and increase the risk of inconsistency in the resulting process models. Moreover, concerns are tightly coupled with individual business processes and could not be reused. As a result, models of business processes become more complex, less reusable and more costly to design and maintain [30].

The Aspect Oriented Paradigm addresses these problems by separating different concerns from the main process. Concerns are captured in terms of *aspect*s associated with a business process and thus can be handled outside the main process. There are various works (e.g [7, 13, 14, 18]), which provide means for aspect oriented business process modelling. Aspect Oriented Business Process Modeling Notation (AO4BPMN) [13] is one such approach that defines the terminology and suggests a notation based on BPMN for modelling processes according to the aspect oriented principle.

Let's consider an example of a business process involving different concerns. Fig. 1 describes a simplified version of a `Transfer Money Process` in the banking domain using BPMN[3]. First, a customer fills in information. Next, if the money is transferred to the customer's own account, the transfer is performed straight away; otherwise, the transaction needs to be signed beforehand. Finally, the transaction is archived. The `Sign Transaction` activity is part of the *security aspect*, and the `Archive Information` activity is part of the *logging aspect*. These aspects describe different concerns related to the `Transfer` activity.

Fig. 2 depicts the above process using AO4BPMN [13]. The concerns are removed from the main process and modelled separately through aspects. Hence, the main process contains only the `Fill Information` and `Transfer` activities. The two additional models annotated with an `Aspect` label are called *aspect models*. They capture the `Logging Aspect` and `Security Aspect`, respectively. An aspect consists of one or more *advices*, which are specified by individual process models annotated with an `Advice` label. An advice captures a specific part of a concern under a certain condition called *pointcut*. A pointcut indicates when and where the advice should be integrated with the main process. The possible points of integrations are called *join points*. A join point can be related to an aspect via a pointcut, and in such case, it is called an *advised joint point*. In AO4BPMN, join points are activities. A pointcut condition, on the one hand, is captured in an annotation associated with an advised joint point activity in the main process, and on the other hand, is specified in a data object as input to the corresponding advice process model. In Fig. 2, the `Transfer` activity is an example of an advised joint point with two pointcuts: one referring to the advice related to information archiving in the `Logging Aspect`, the other to the advice related to transaction signing in the `Security Aspect`.

[3] For simplicity, we omit pools/lanes in this process model.

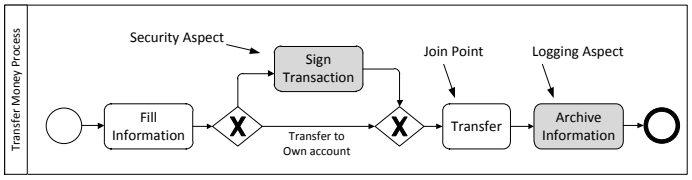


Fig. 1. Transfer Money Process model in BPMN

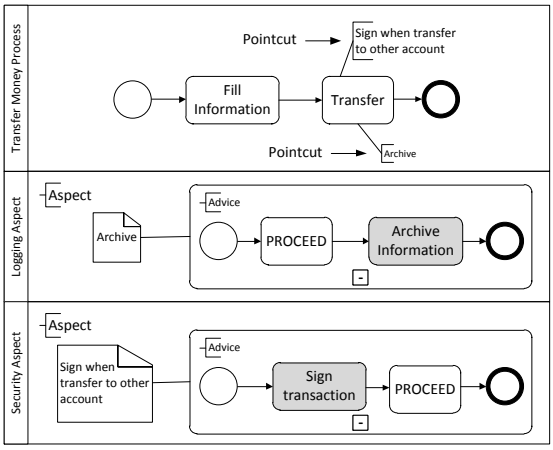


Fig. 2. Transfer Money Process model in AO4BPMN

Next, a `PROCEED` activity within an advice model acts as a "placeholder". This placeholder is for carrying out the relevant advised joint point activity during the execution of the advice. As such, the position of a `PROCEED` activity determines how the execution of an advice interleaves with the related advised joint point activity. There are three scenarios: an advice occurring *before*, *after* or *around* an advised joint point. For example, in Fig. 2, the `Archive Information` activity occurs after the `Transfer` activity, while the `Sign Transaction` activity occurs before the `Transfer` activity. It is possible that an advice does not have a `PROCEED` activity. Such advice is called an *implicit advice*, and is executed *before* the related advised joint point activity.

Comparing the process models in Fig. 1 and Fig. 2 shows that aspect oriented process modelling increases reusability because an aspect can be related to different activities and even different processes. It also makes the maintenance of process models easier, since any change to a concern would only affect the relevant aspect. Finally, it reduces the complexity of process models that capture the core processes.

3 Overview of the Solution

In the area of BPM, the existing aspect oriented modularisation approaches, such as AO4BPMN, only support process modelling. These models should be

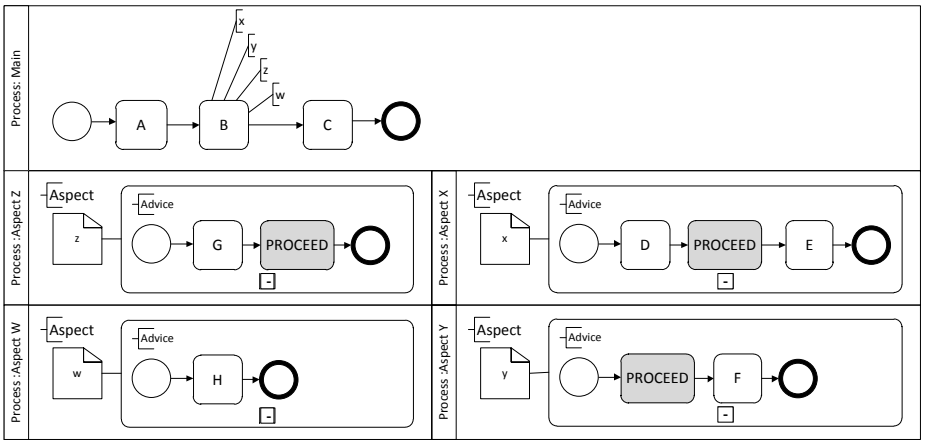


Fig. 3. An abstract example of an aspect oriented process model

weaved in order to be executable. The weaving can be performed at design time or at runtime, which are called static or dynamic weaving correspondingly. Static weaving does not resilient to change, since for every change the process model should be weaved and loaded into BPMS. However, dynamic weaving is the approach that is flexible and solve such problem. Dynamic weaving of aspects are investigated in different areas like programming(e.g. [27, 26]), service composition [11] and etc. However, it is still an open issue in BPM area.

In this section, we propose a generic solution in form of a so-called *Aspect Service*, which extends current BPMSs with runtime support to aspect oriented modularization.

According to the principle of weaving, the aspects and their advices are to be executed together with the main process, and synchronisations are to be made at the `Proceed` placeholder as well as at the end of each advice. To explain such requirements of weaving in more detail, we use an abstract example of an aspect oriented process model shown in Fig. 3. There is a main process with four aspects, which are associated to one of the activities (activity B) in the process, and each aspect contains a single advice. Based on the fact that aspect Y has an *after* advice, aspect Z has a *before* advice, aspect X has an *around* advice, and aspect W has an implicit advice, it can be determined when activity B should take place together with the four advices. Moreover, activity C, which is the activity that follows activity B, in the main process, cannot occur until the executions of *all* the advices associated with activity B are completed. Hence, the valid execution sequence of activities in the process in Fig. 3 will be A, followed by D, G and H in parallel, then B, followed by E and F in parallel, and finally C. Using regular expression this can be written as $A(D||G||H)B(E||F)C$.

At runtime, the enactment of business processes, including executions of activities in the main process and those in the associated aspects, is managed through a BPMS. The Aspect Service controls and coordinates the interactions between (the advices in) the aspects and the main process. We propose a generic approach to support dynamic weaving of aspects during process enactment. It

consists of the following four steps. Note that for an implicit advice, a pre-processing is required which adds an (empty) Proceed placeholder to the end of the advice, and as such an implicit advice is treated as before advice during dynamic weaving.

1 *Launching*: *before executing an advised joint point, the Aspect Service shall launch all valid advices associated with that joint point.* Each valid advice is determined by the Aspect Service through evaluation of the corresponding pointcut condition. If the pointcut condition holds, the Aspect Service will initiate one instance for that valid advice.

2 *Pausing*: *the Aspect Service shall pause the execution of an advice when reaching the Proceed activity in the advice and at the same time enable the execution of the corresponding advised joint point in the main process.* It is possible that multiple advices exist for one advised join point, in which case, the `Proceed` activities in these advices should be synchronized to ensure a single execution of the advised joint point.

3 *Resuming*: *when the execution of the advised join point is completed, the Aspect Service shall resume the enactment of those advice instances that are interrupted by the execution of the advised joint point.* Note that for each launched advice (regardless it being a before, after, or around advice), the control of execution will be returned to the corresponding advice instance after the enactment of the advised joint point.

4 *Finalizing*: *the Aspect Service shall complete the executions of all the launched advice instances before the enactment of the main process can continue.* Only when the executions of all the launched advice instances finish, the control will be returned to the main process to continue its enactment (to subsequent activities enabled after the advised joint point). This signals the end of the weaving of aspects associated with that advised joint point.

Fig. 4 illustrates the dynamic weaving of the aspects with the main process using the example shown in Fig. 3. To reflect runtime nature of weaving, we use the notation of Yet Another Workflow Language (YAWL). YAWL is based on Petri nets but extended with advanced control-flow constructs to facilitate workflow modelling. In the left-hand side of Fig. 4, there are four YAWL nets capturing the main process and the three advice processes, respectively. Between each advice net and the main net, there are highlighted annotations capturing the above four-phased weaving approach. For illustration purpose, the overall behaviour resulting from the weaving of three aspects with the main process of the example in Fig. 3 is specified as the YAWL net in the right-hand side of Fig. 4. Tasks that belong to the same advice share the same graphical annotation.

Next, we look into specific states and state changes during process enactment to elaborate the above weaving approach. At runtime, an instance of an activity is called work item. Russell et al. define a general lifecycle of a work item, which describes the possible states and relation between these states during the execution of a work item [29] (See Fig. 5). In each state, different information is

available concerning different perspectives . For example, the data of the resource perspective is not available when a work item is in the `Created` state. Instead, it is available when the work item is in the `Started` state.

To separate cross-cutting concerns, we need to have information regarding different perspectives. The states in which a work item has all information are `Started`, `Suspended`, `Completed` and `Failed` states. As it can be seen in Fig. 5, a work item can be alternate between `Started` state and `Suspended` state. Hence, these states are central for the weaving of advice to a main process. Therefore, the following states changes can be defined for instances of activities in both main and advices models during the weaving of aspects.

For *Launching*, the state of an advised join point shall be changed from `started` to `Suspended`. Then, all advices shall be launched. For *Pausing*, the `Suspended` state of the advised join point should be changed to `Started`, i.e. ready to be executed. For *Resuming*, the remainder activities of the main process should not be executed, and the advised join point state remains in `Completed`. The solution is to prevent instances of other activities to be `Completed`. Therefore, other instances of all other activities should be changed to `Suspended` if they reach to the `Started` state. For *Finalizing*, the suspended work items in previous step should be changed to `Started` again.

During all these steps, the data should also be synchronised between the main process and its advices. In case several advices operate on the same data simultaneously (see for example activities D and G in Fig. 4), the last event will overwrite the data that are stored by the work items of the advised joint point, that has already been completed. In the next section, we describe the CPN model that shows the operational semantics of the weaving at runtime.

4 Formal Specification

The formalisation of the Aspect Service is specified using hierarchical Coloured Petri Nets. The solution is a three-level model. The top-level module captures the behaviour of the initiation of the service (see Fig. 6). The second level captures the weaving behaviour (see Fig. 7). This model contains four modules capturing the requirements related to weaving described in the previous section. It also

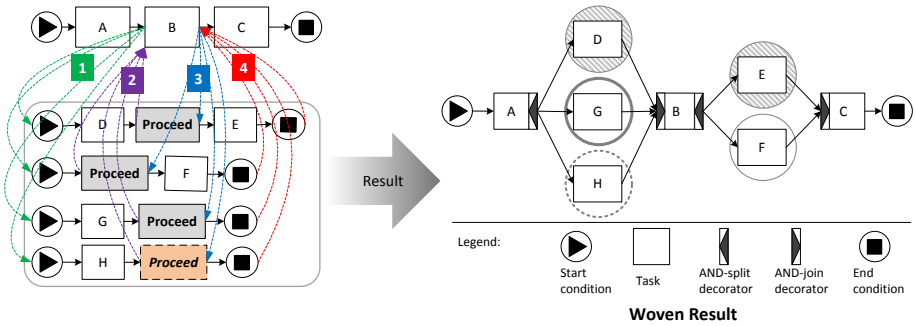


Fig. 4. Dynamic weaving of aspects using the example in Fig. 3

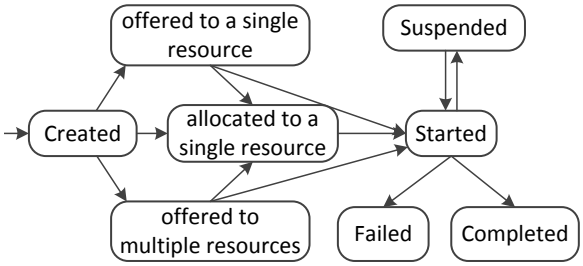


Fig. 5. Work item Lifecycle [29]

contains a module for communicating with the BPMS and performing actions for data persistence, which is needed for the weaving. These five modules constitute the third level of the CPN Model. The model defines 57 colour sets and 33 functions. We re-used some of the colour sets, variables and functions from the Worklet Service CPN model [5]. A preliminary version of the CPN model from our previous work is reported in [19]. In that version, the semantic was not developed based on workitem lifecycle; while, this version is refined to be compatible with the lifecycle. This compatibility makes the semantic general for all workflow management systems.

The interaction of the Aspect Service and a BPMS is realized through passing a number of messages. These messages are named constraints and commands. *Constraints* are the messages raised by the BPMS, and *Commands* are the messages invoked by the Aspect Service.

We used standard constraints and messages according to the BPMS reference model defined by the Workflow Management Coalition (WfMC). For concrete names of messages, we adopted those in the YAWL system which is known

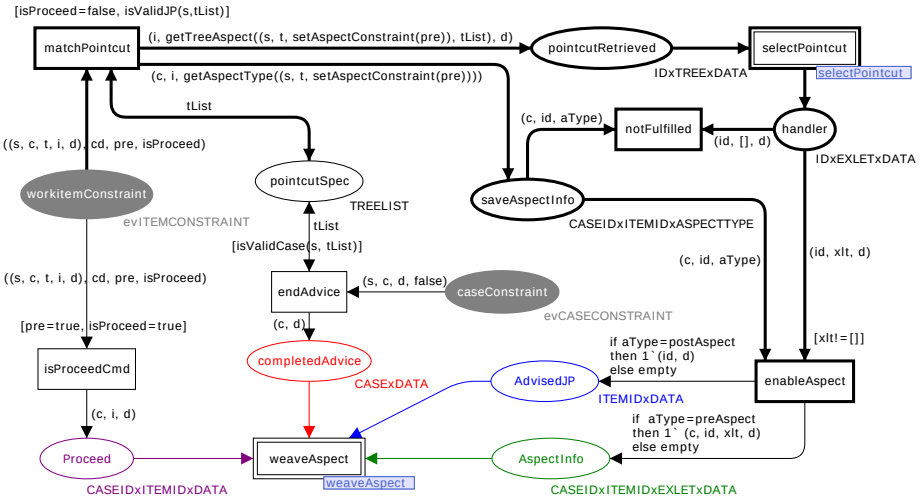


Fig. 6. CPN: Aspect Service

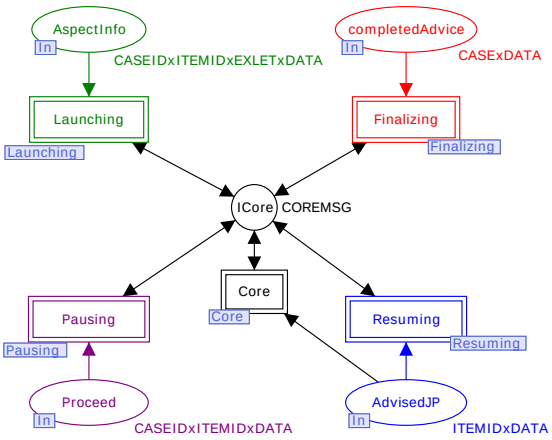


Fig. 7. CPN: weaveAspect

as conforming to the WfMC's reference model. Hence, the solution is general and can be adapted to any BPMS. The constraint messages are *WorkitemConstraint* and *CaseConstraint*. The raising of one of these messages is signified in the CPN model in Fig. 6 as a token arriving in the `workitemConstraint` or `caseConstraint` places correspondingly (the places are highlighted in the figure). In other words, these places are the starting points of the net.

As it can be seen in Fig. 6, the service recognizes if the WorkitemConstraint is related to a normal activity or to a `Proceed` activity (see the `matchPointcut` and `isProceedCmd` transitions). If it is a *normal activity*, the `matchPointcut` investigates whether a pointcut is defined for the activity or not. If yes, the pointcut is evaluated to see which advices should be launched using `selectPointcut` subnet. If the pointcut is fulfilled, the `enableAspect` transition is enabled, else the `notFulfilled` transition is enabled. The result leads to *Launching* if the advised join point is just started (the `enableAspect` transition produces a token in `AspectInfo` place). However, if the advised join point is completed, the *Resuming* should be started (the `enableAspect` transition produces a token in `AdvisedJP` place). If it is a *Proceed activity*, no Pointcut is needed to be checked (the `isProceedCmd` transition produces a token in `Proceed` place). As a result, the *Pausing* can be started. The *Finalizing* can be started if all advices are finished. This condition is checked using `endAdvice` transition, which produces a token in `completedAdvice` place if a token representing the end of a launched advice appears in `caseConstraint`.

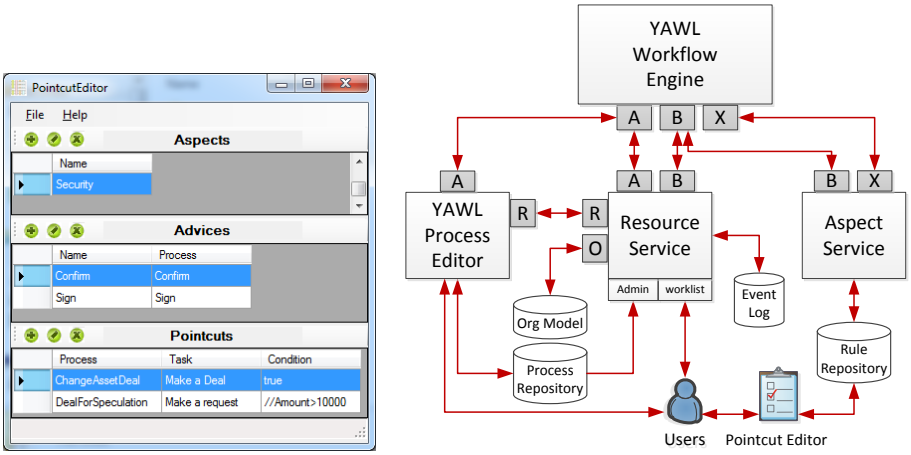
The net in Fig. 6 shows how messages received from the BPMS should be processed to enable weaving. The weaving is described by the `weaveAspect` sub-net shown in Fig 7. The `weaveAspect` net contains five subnets such as `Launching`, `Pausing`, `Resuming`, `Finalizing` and `Core`. The first four subnets fulfils the four weaving requirements, and the last one persists the data which are required to perform weaving.

The CPN model allowed us to verify the design of the Aspect Service using state space analysis. This analysis showed that the nets were free of deadlocks. Moreover, the Strongly Connected Component (SCC) graph of the model has the same number of nodes and arcs as the corresponding state space. This indicates that there is no cycle in the CPN model, which is expected since a weaving process taken into account within the scope of current work is free of cycles.

Due to space consideration the rest of the CPN model is not described here. Full details of the model with definition of the colour sets, variables, functions and analysis are publically available and can be downloaded[4].

5 Implementation

Using the formal CPN specification of our proposed solution for dynamic weaving of aspects for process enactment, we implemented two artefacts, a Pointcut Editor and an Aspect Service[5], in the framework of the YAWL system[6]. We chose YAWL because: 1) it provides support to the full workitem life cycle; 2) it has formal foundation; and 3) it is open-source and based on Service Oriented Architecture [2, 3].



(a) Pointcut Editor (b) Aspect Service Architecture

Fig. 8. Implemented artefacts to enable dynamic weaving

The Pointcut Editor (see a screenshot of the GUI in Fig. 8(a)) enables the definition of aspects, advices and pointcuts. Each aspect can have several advices, and each advice can have several pointcuts. These pointcuts consist of the name of the process and activity for which the advice should be weaved. The pointcut

[4] http://www.aobpm.com

[5] The artefacts, with examples and case studies, can be downloaded from http://www.aobpm.com

[6] An open source BPMS, see `http://www.yawlfoundation.org`

also includes a condition. The condition is used to define data constraints, which controls the enactment of an advice (e.g. transfer to non own account). If the condition is fulfilled, the advice will be weaved. The condition can be written using XPath language.

Fig. 8(b) shows the architecture of the Aspect Service and its relation to the BPMS. The service is connected to the YAWL Engine through two interfaces, i.e. B and X. Interface X and B are used to capture case and workitem level events correspondingly. The service also reads the rules (composed by the Pointcut Editor) from the Rule Repository. The rules specify which events should be captured. During implementation, the example in Fig. 3 was used for testing, since it contains all combinations of advice types.

6 Case Study

In this section, we apply the aspect oriented approach to a real case from the financial domain[7]. The case demonstrates how the aspect oriented modularization can be used to capture cross-cutting concerns in two banking process models and thus enactment of the two aspect oriented process models using the implemented artefacts in the previous section.

These processes were modelled in a traditional way first (see Fig. 9). Then, cross-cutting concerns were separated from them by applying AO4BPMN (see Fig. 10). Afterwards, the AO4BPMN model (see Fig. 10) was manually converted to a YAWL model for execution in the YAWL system. The `Pointcut Editor` was used to define pointcuts, and the `Aspect Service` was used to enact processes.

The banking case was selected due to our previous knowledge in that domain. To choose appropriate processes, i.e. fairly simple yet representative processes, we conducted an interview with a domain expert from a bank. For the confidentiality reason, the bank asked to remain anonymous. Two processes were selected, i.e. the *Deal for speculation* process and the *Change Asset Deal* process. Detailed information about the processes was derived through a follow-up interview with the same domain expert.

The goal of the *Deal for speculation* process is to make a profit; however, sometimes money is lost. Hence, there is a limit on the amount of money that a junior and a chief dealer can trade in a deal. If a junior dealer wants to use a higher amount, an approval from his chief is needed. If the amount exceeds the limit of the chief dealer, an approval from the general manager (GM) is also needed. This approval needs to be archived by the `Office Employee`. If an approval is obtained or a deal is within one's limit, a *junior dealer* opens a position, makes the deal, and fills in a deal slip. Next, both a *chief dealer* and the *general manager* sign the deal slip, after which the deal slip is archived. After this, two parallel sets of activities are performed. On one hand, the dealt amount of money is sent to the external partner of the deal. For this, first an *employee of the Swift department* provides a swift draft for sending the money. Then, for security purposes, the *dealer*, *chief dealer* and *general manager* sign the swift draft. Finally, an *employee of the Swift department* sends out the swift. In parallel, the dealt

[7] Translation of the banking terminology to English is done by the authors.

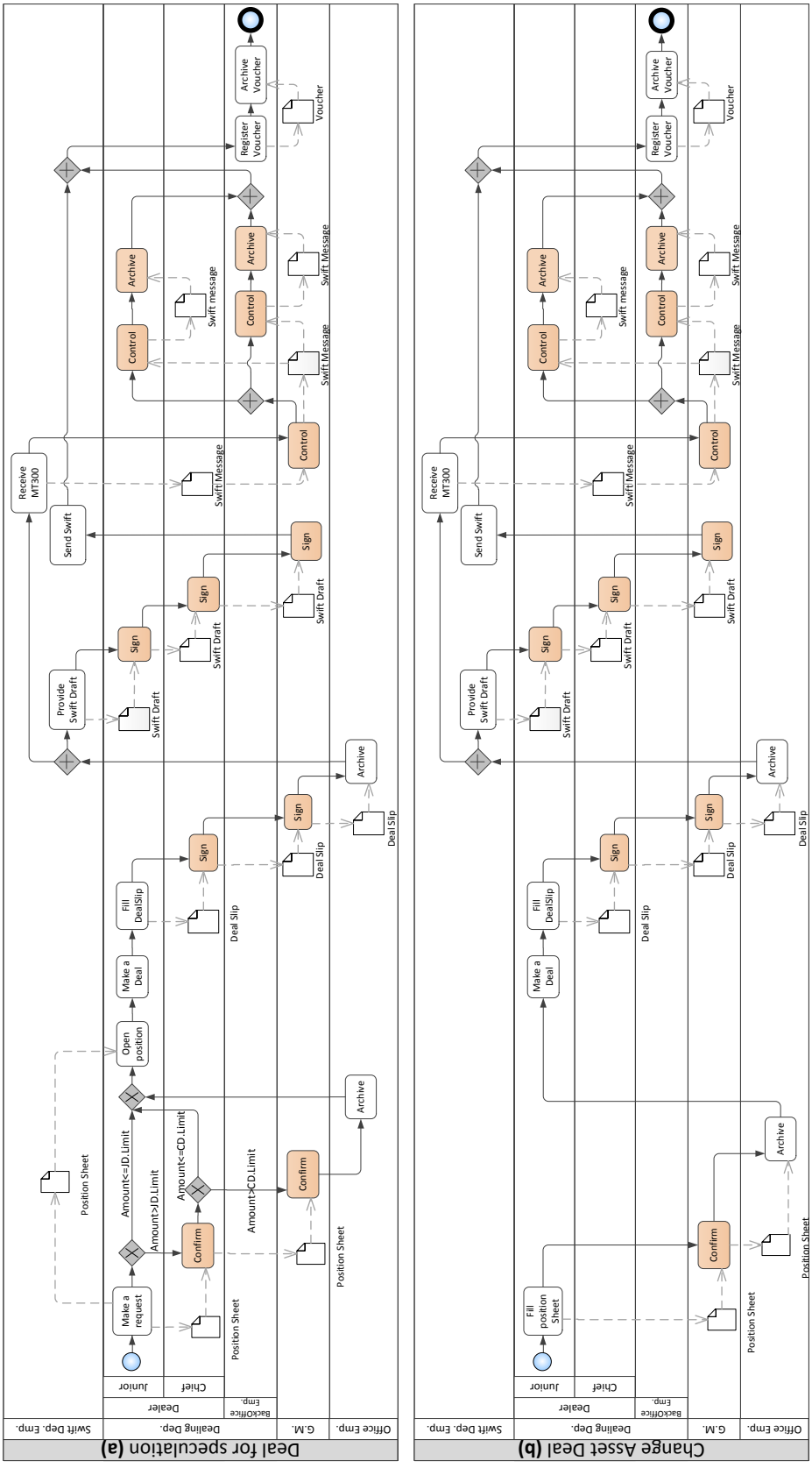


Fig. 9. The AS-IS case study processes

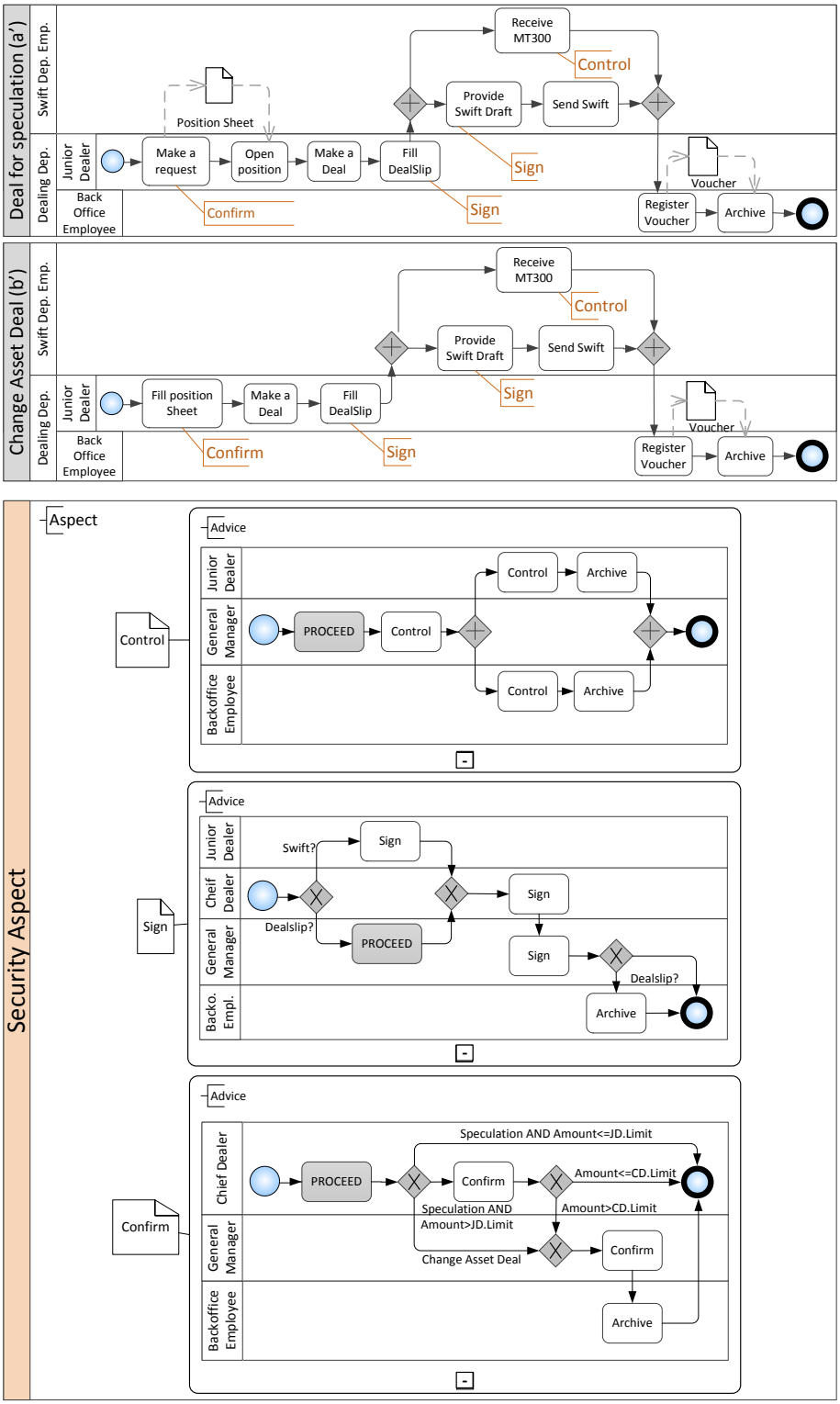


Fig. 10. The TO-BE case study processes

amount of money should be received. This part starts when an *employee of the Swift department* receives an MT300 swift message. The *employee* sends this message to the general manager to control. The *general manager* makes an order to the Back office department and to the dealer to control the swift message. These orders are issued separately, according to the security policy at the bank. The results from both controls are archived separately. When the deal is made, a *back office employee* registers a voucher in the accounting system. The process ends with archiving the voucher. Fig. 10 shows the process modeled with the aspect oriented approach. In this model, the security concern is separated from the main process and captured in the Security Aspect with a number of advices, i.e. `Confirm`, `Control` and `Sign`.

The goal of the *Change Asset Deal* process is to change some asset of the bank from one currency to another. The process starts by `Backoffice Employee` who fills in the position sheet. The `General Manager` confirms the position sheet, and `Office Employee` archives it. Next, the `Junior Dealer` makes the deal and fills in dealslip. The rest of the process is the same as *Deal for speculation* process.

These processes are implemented with limited information from the data perspective. This limitation does not affect on validating the artefacts, since the current approach focuses on the execution of control-flow perspective.

Below we summarise our experiences from carrying out this case study.

- *The aspect oriented approach truly enables separation of concerns.* Separation of concerns like security or privacy increases the reusability, since they are defined once (at organisation level) and applied across the organisation where needed. It also *facilitates the maintenance* of a system. If a policy is changed, the changes are reflected in one model rather than in all business processes utilising it. In our case, if the control routines at the bank are increased, the updates are reflected in the corresponding advice(s) instead of in the processes implementing them.
- *Aspect oriented decomposition decreases the complexity of process models through decreasing the overall size of models.* Hence, communicating models to business users is expected to be easier [23]. The sizes of process models are decreased both in deal processes and in overall(considering advice activities). The *deal processes* in Fig. 10 contain only half the number of activities compared to the original (not aspect oriented) model. The overall size of the set of models is also decreased since the repetitive parts in both processes are modelled once. The total size of the models applying the aspect oriented approach, were more than 25 percent less than the models not using aspect oriented decomposition. Furthermore, swimlanes which represents people who are only involved in security aspect are disappeared from the main processes, i.e. `General Manager` and `Chief Dealer`. This also adds the readability of models. It should be noted that the process models in the case study are fairly small, so the expected effect of applying aspect oriented modularization on larger process models will be even more significant.
- *Aspect oriented modelling documents additional knowledge about the business processes.* This knowledge specifies the relation between cross-cutting concerns and activities. For example, in this case study, we can see that the `Security` aspect (more precisely the `Sign` advice) is associated to the `Send`

`Swift` activity. This information is not captured in the process modelled with traditional approaches, where we cannot interpret whether `Provide Swift Draft` or `Send Swift` activity is related to the security aspect.

- *Guidelines on how to apply aspect oriented decomposition are needed.* Sometimes different design choices are possible. For instance, the `Archive` activities in advices might also be considered as different aspects, i.e. logging. Guidelines supporting such design choices would help business process analysts in applying the aspect oriented approach.
- *The possibility to define the sequential order of advices associated to the same activity should be offered.* The approach offered by Charfi et al [13] or Cappelli et al [7] do not consider the definition of precedence for advices. This limitation enforces us to dismiss separation of some aspects from the main process or merge aspects together (like having `Archive` and `Sign` activities in `Sign` advice in our case). The first solution limits the aspect oriented modelling to separate all cross-cutting concerns from process models; while, the second one makes aspects more coupled together, which decreases the reusability of them for different process models. Hence, the Aspect Service should support the definition of precedence between advices [18]. This is to be considered as an extension to our current work.
- *The advice type should be explicitly defined in pointcut rather implicitly using a `Proceed` placeholder.* This study shows that the way that AO4BPMN proposes the definition of advice types can reduce reusability of advices. For example, a sign advice which is defined as *before* advice (using the `Proceed` placeholder) cannot be used as *after* advice later. The solution is to define the advice type in pointcut to increase the reusability of advices for both scenarios, as what is proposed in [13] and [18].

7 Related Work

To support the Aspect Oriented paradigm two components are needed: *decomposition* for capturing separation of concerns, and integration i.e. the *weaving* of aspects with processes. In the process modelling area, there are some attempts for process decomposition, e.g. [9, 13, 18, 22, 24, 31]. Despite these numerous attempts, we could not find any work which shows how *weaving should be performed* in BPM area. The weaving can be performed in design time or run-time, namely static or dynamic weaving correspondingly. Static weaving suffers from lack of flexibility, since it needs models to be weaved and uploaded into business process for every change. In contrast, dynamic weaving covers this lack, and it provides flexibility to change aspects at runtime [25]. Therefore, in the work presented here, we elaborate on the *Dynamic Weaving* for BPM. The weaving is inspired by the work on weaving in programming e.g. [8, 16, 20, 21].

Moreover, it should be mentioned that there is one implementation of weaving for service orchestration, namely AO4BPEL [12]. AO4BPEL is an extension to the Business Process Execution Language (BPEL) to support aspect orientation. This extension is defined based on soap message lifecycle [10]. This means that the BPEL4People activity lifecycle is not considered at designing this extension [6], which makes the approach specific to service decomposition. Such a

limit disables AO4BPEL to address the need of separation of cross-cutting concerns in BPM area. This need is even reflected by Charfi A. where he mentions "These security concerns will not be shown in BPEL code because BPEL does not support human participants. There is however, a recent proposal for such an extension". To consider the proposal (BPEL4People), the solution (AO4BPEL) should be changed to comply with BPEL4People activity lifecycle. However, to the best of our knowledge, no research has been done to address this issue.

Furthermore, AO4BPEL cannot be used to study the needs of separation of concerns for other business process perspectives since BPEL does not support all of business process perspectives. Such a need can be exemplified as the situation where a senior employee in the bank shall confirm all activities of newly employed clerk at the first week. This separation needs definition of pointcuts to be specific for resource perspective. Such a separation cannot be investigated by AO4BPEL since it is not developed based on workitem or BPEL4People activity lifecycles [17].

8 Conclusions and Future Work

In this paper, we presented a generic solution to address how the weaving of aspects to business processes can be done. The solution is designed and implemented in form of a service, namely the Aspect Service, which extends a BPMS to support enactment of aspect oriented business process models. We provided a formalisation of the Aspect Service using CPNs and verified the soundness of the design of this service (using state space analysis). The Aspect Service is implemented in YAWL based on defined semantic. The implemented service shows that aspect oriented business process modelling increases the reusability, reduces the complexity and facilitates the maintenance of process models. The artefact is also inspected through implementing two processes of a case study from banking domain. The implementation not only shows the relevancy of the artefact to solve the separation of cross-cutting concerns, but it also reveals limitations of current aspect oriented modeling techniques. Therefore, a direction for future work is defined based on real application of aspect orientated business process modeling and enactment.

The solution is currently limited to weaving advices in which the Proceed placeholder is enabled only once. This means Proceed cannot be included in loops. Moreover, if several Proceed placeholders are defined within the same advice, care must be taken that only one of them is enabled during the execution of the advice (e.g. as a result of an XOR split). The impact of these limitations, i.e. how frequent such scenarios occur in real life, needs to be studied further.

Other directions for future work include: (i) a comparison of Aspect Orientation in the programming and BPM areas. Such comparison would fortify Aspect Oriented BPM, as the Aspect Orientation is more mature in the programming area; (ii) a definition of a pointcut language which captures other business process perspectives such as the resource perspective; (iii) an investigation on how the resource patterns [29], e.g., separation of duties and retain familiar, should be captured in orthogonal modularization; (iv) an extension to the semantic and implementation of Aspect Service to support weaving of ordered aspects in BPM

area; (v) an investigation on the possibility to define nested aspects, i.e. an aspect that is related to other aspects; and (vi) extending the implementation of Aspect Service to support other WfMSs as well.

References

1. W.M.P. van der Aalst. Three Good reasons for Using a Petri-net-based Workflow Management System. In T. Wakayama et al., editor, *Information and Process Integration in Enterprises*, volume 428, pages 161–182. Springer US, 1998.
2. W.M.P. van der Aalst, L. Aldred, M. Dumas, and A.H.M. ter Hofstede. Design and implementation of the yawl system. In A. Persson and J. Stirna, editors, *CAiSE*, volume 3084 of *LNCS*, pages 281–305. Springer, Springer, 2004.
3. W.M.P. van der Aalst and A.H.M. ter Hofstede. Yawl: yet another workflow language. *Information Systems*, 30(4):245–275, 2005.
4. W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske. Business Process Management: A Survey. In M. Weske, editor, *BPM Conference*, volume 2678 of *LNCS*, pages 1019–1019. Springer, 2003.
5. M.J. Adams. *Facilitating Dynamic Flexibility and Exception Handling for Workflows*. PhD thesis, Faculty of Information Technology, Queensland University of Technology, November 2007.
6. A. et al. Agrawal. WS-BPEL extension for people (BPEL4People), version 1.0. *OASIS Cover Pages: http://xml. coverpages. org/BPEL4People-V1-200706. pdf*, 2007.
7. C. Cappelli at. all. Reflections on the modularity of business process models: The case for introducing the aspect-oriented paradigm. *BPM Journal*, 16:662–687, 1995.
8. N. Belblidia and M. Debbabi. Formalizing AspectJ Weaving for Static Pointcuts. In *SEFM*, pages 50–59. IEEE Computer Society, 2006.
9. C. Cappelli, J.C.S.P Leite, T. Batista, and L. Silva. An aspect-oriented approach to business process modeling. In *Proceedings of the 15th workshop on Early aspects*, EA '09, pages 7–12. ACM, 2009.
10. A. Charfi. *Aspect-oriented workflow languages: AO4BPEL and applications*. PhD thesis, der Technischen Universitat Darmstadt, Darmstadt, November 2007.
11. A. Charfi and M. Mezini. Aspect-Oriented Web Service Composition with AO4BPEL. In L. Zhang and M. Jeckle, editors, *Web Services*, volume 3250 of *LNCS*, pages 168–182. Springer Berlin Heidelberg, 2004.
12. A. Charfi and M. Mezini. AO4BPEL: An Aspect-oriented Extension to BPEL. *World Wide Web*, 10:309–344, 2007.
13. A. Charfi, H. Müller, and M. Mezini. Aspect-Oriented Business Process Modeling with AO4BPMN. In T. Khne et al., editor, *Modelling Foundations and Applications*, volume 6138 of *LNCS*, pages 48–61. Springer, 2010.
14. C. Di Francescomarino. Supporting Documentation and Evolution of Crosscutting Concerns in Business Processes. In H.R. Motahari Nezhad et al., editor, *ICSOC PhD Symposium*, volume 421 of *CEUR Workshop Proceedings*, 2008.
15. V. Gruhn and R. Laue. Reducing the cognitive complexity of business process models. In *IEEE ICCI*, pages 339–345, 2009.
16. W.-M. Ho, J.-M. Jézéquel, F. Pennaneac'h, and N. Plouzeau. A Toolkit for Weaving Aspect Oriented UML Designs. In *AOSD*, pages 99–105, 2002.
17. A. Jalali and P. Johannesson. Multi-Perspective Business Process Monitoring. In S. Nurcan et al., editor, *Enterprise, Business-Process and Information Systems Modeling*, volume 147 of *LNBIP*, pages 199–213. Springer Berlin Heidelberg, 2013.

18. A. Jalali, P. Wohed, and C. Ouyang. Aspect oriented business process modelling with precedence. In J. Mendling et al., editor, *BPMN*, volume 125 of *LNCS*, pages 23–37. Springer, 2012.
19. A. Jalali, P. Wohed, and C. Ouyang. Operational semantics of aspects in business process management. In P. Herrero et al., editor, *OTM 2012 Workshops*, volume 7567, pages 649–653. Springer, 2012.
20. J.-M. Jézéquel. Model Driven Design and Aspect Weaving. *Software and System Modeling*, 7(2):209–218, 2008.
21. J. Klein, F. Fleurey, and J.-M. Jézéquel. Weaving Multiple Aspects in Sequence Diagrams. In A. Rashid and M. Aksit, editors, *Transactions on Aspect-Oriented Software Development III*, volume 3 of *LNCS*, pages 167–199. 2007.
22. I. Machado, R. Bonifácio, V. Alves, L. Turnes, and G. Machado. Managing variability in business processes: an aspect-oriented approach. In *Proceedings of the 2011 international workshop on Early aspects*, EA '11, pages 25–30. ACM, 2011.
23. J. Mendling, H. Reijers, and J. Cardoso. What Makes Process Models Understandable? In G. Alonso et al., editor, *BPM Conference*, volume 4714 of *LNCS*, pages 48–63. Springer, 2007.
24. A. Perin-Souza, C. Cappelli, F.M. Santoro, L.G. Azevedo, J.C.S. do Prado Leite, and T.V. Batista. Service identification in aspect-oriented business process models. In *SOSE*, pages 164–174, 2011.
25. M. Pinto, L. Fuentes, M.E. Fayad, and J.M. Troya. Separation of coordination in a dynamic aspect oriented framework. In *Proceedings of the 1st international conference on Aspect-oriented software development*, pages 134–140. ACM, 2002.
26. A. Popovici, G. Alonso, and T. Gross. Just-In-Time Aspects: Efficient Dynamic Weaving for Java. In *In Proceedings of the 2nd international conference on Aspect-oriented software development*, pages 100–109. ACM Press, 2003.
27. A. Popovici, T. Gross, and G. Alonso. Dynamic weaving for aspect-oriented programming. In *Proceedings of the 1st international conference on Aspect-oriented software development*, AOSD '02, pages 141–147. ACM, 2002.
28. M. La Rosa, P. Wohed, J. Mendling, A.H.M. ter Hofstede, H.A. Reijers, and W.M.P. van der Aalst. Managing Process Model Complexity Via Abstract Syntax Modifications. *IEEE Trans. Industrial Informatics*, 7(4):614–629, 2011.
29. N. Russell, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In O. Pastor et al., editor, *CAiSE*, volume 3520 of *LNCS*, pages 216–232. Springer, 2005.
30. C. SantÁnna, A. Garcia, Ch. Chavez, C. Lucena, and A. von Staa. On the reuse and maintenance of aspect-oriented software: An assessment framework. In *Proceedings of Brazilian Symposium on Software Engineering*, pages 19–34, 2003.
31. N. Santos, F. Jack, S. do Prado Leite, J. Cesar, C. Claudia, T. Batista, and F.M. Santoro. Using goals to identify aspects in business process models. In *Proc. of the 2011 international workshop on Early aspects*, EA '11, pages 19–23. ACM, 2011.
32. A. Streit, B. Pham, and R. Brown. Visualization Support for Managing Large Business Process Specifications. In W. van der Aalst et al., editor, *BPM*, volume 3649 of *LNCS*, pages 205–219. Springer, 2005.
33. B. Weber, M. Reichert, J. Mendling, and H.A. Reijers. Refactoring large process model repositories. *Computers in Industry*, 62(5):467–486, 2011.
34. M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.